JOURNAL OF GUIDANCE, CONTROL, AND DYNAMICS
Vol. 24, No. 4, July–August 2001

Stable Neuro-Flight-Controller Using Fully Tuned Radial Basis Function Neural Networks

Yan Li,* Narasimhan Sundararajan,† and Paramasivan Saratchandran‡
Nanyang Technological University, Singapore 639798, Republic of Singapore

A flight control scheme in which a radial basis function network (RBFN) aids a conventional controller has been developed. The RBFN controller, consisting of variable Gaussian functions, uses only online learning to represent the local inverse dynamics of the aircraft system. With a Lyapunov synthesis approach, a tuning rule for updating all of the parameters of the RBFN (including centers, widths, as well as the weights of the output layer) is derived, which extends Gomi and Kawato's strategy, where only the weights were adaptable. (Gomi, H., and Kawato, M., "Neural Network Control for a Closed-Loop System Using Feedback-Error Learning," *Neural Networks*, Vol. 6, No. 7, 1993, pp. 933–946). The proposed tuning rule guarantees the convergence of the overall system and greatly improves the tracking accuracy. Simulation studies using an F8 aircraft longitudinal model illustrate the superior performance of the proposed scheme. The simulation studies further indicate that the results can be extended to a dynamic RBFN in which the hidden neurons can be added/pruned, thus producing a more compact network structure.

I. Introduction

BECAUSE of their powerful ability in approximating nonlinear functions, control laws and design methods incorporating artificial neural networks have been extensively studied. In the area of flight control, Calise and Rysdyk summarized some current research efforts toward applying neural network (NN) technology for flight control system design.[1] Results on nonlinear flight control are further presented in Refs. 2 and 3, where the advantages of recent stability proofs for multi-layer-perceptron NN are utilized. It has been shown that NN with online learning can adapt to aircraft dynamics that are poorly known or rapidly changing.

NN with online learning capability for flight control applications was reported in Ref. 4, where Sadhukhan and Feteih presented an exact inverse neurocontroller with full-state feedback for an F8 aircraft. The objective of the controller in an autopilot mode was to closely track the pilot's throttle and pitch rate commands. Napolitano et al.[5] studied the performance of various NN-based designs of autopilot systems for implementing a variety of flight control functions. In most of these applications, a feedforward network with a back propagation (BP) learning algorithm or its extentions has been the main paradigm. It is known that, when using a feedforward network and the BP algorithm, problems arise with local minima and saddle points, and the algorithm itself has a very slow convergence rate. Furthermore, in Ref. 6, limitations of using NN in real world adaptive control are analyzed, and the future research effort is also pointed out.

Since the early 1990s, radial basis function networks (RBFN) with Gaussian function have been widely used as the basic structure of NN in nonlinear control,[7,8] due to their good global generalization ability and a simple network structure that avoids unnecessary and lengthy calculations.[9] For online implementation, Sanner and Slotine[10] developed a direct adaptive tracking control architecture using Gaussian RBFN to adaptively compensate for plant nonlinearities. Gomi and Kawato[7] proposed a feedback-error-learning control strategy, where a Gaussian RBFN is used for online learning of the inverse dynamics of the system. Recently, Polycarpou[11] and Polycarpou and Mears[12] have also presented a control design approach for a class of nonlinear plants, where an adaptive bounding technique is employed to handle the unknown network reconstruction error. In those schemes, the adaptive tuning rules are derived based on a Lyapunov synthesis approach, to guarantee the closed-loop stability. However, in aircraft flight control, there are only limited papers that explore the application of RBFN. In Ref. 13, Singh and Wells used an RBFN to suppress wing rock for a slender delta wing configuration. Kim and Calise presented the use of RBFN to capture variations in Mach number in Ref. 14, because these variations are difficult to represent with polynomic functions in the transonic region. In all of the aforementioned applications, when implementing the network structure, the number of hidden units, centers, and widths are always fixed, and then the weights of the network are estimated. Practically speaking, it is difficult to choose the centers and widths appropriately, especially for online implementation where the dynamics may be changing. The use of inaccurate centers usually results in the deterioration of performance. To overcome this problem, a large number of hidden neurons need to be recruited, and this, in turn, results in slow online learning.[15,16]

In comparison to conventional approaches, recently, fully tuned RBFN have begun to exhibit their potential for accurate approximation. In a fully tuned RBFN, not only the weights of the output layer, but also the other parameters of the network (like the centers and widths) are updated, so that the nonlinearities of the dynamic system can be captured as quickly as possible.[15,17] It is in this context that this paper attempts to explore the use of the fully tuned RBFN for aircraft control applications. A simple control architecture originating from Gomi and Kawato's feedback-error-learning scheme, incorporating a fully tuned RBFN controller, is proposed. With use of the Lyapunov synthesis approach, a stable tuning rule for updating all of the parameters of the RBFN are derived and the local stability of the overall system along the desired trajectory is guaranteed. In addition, the robustness of the proposed tuning law in the presence of the NN approximation error, as well as the system model error, is also analyzed.

For comparison purposes, the linearized F8 longitudinal aircraft model of Ref. 4 is utilized for simulation studies, and the results demonstrate the efficiency of the proposed method. Note that the proposed scheme can also be applied to control nonlinear aircraft dynamics. The corresponding work is carred out in Ref. 18. Moreover, a dynamic RBFN with growing and pruning (GAP) strategy is investigated, and the results indicate that, with the GAP strategy, the network structure is more parsimonious.

The paper is organized as follows: Section II gives a brief description of the problem, and Sec. III derives the stable adaptive tuning rule for a fully tuned RBFN, where the stability of the overall system is guaranteed. A robustness analysis of the proposed scheme to the approximation error and model error is also presented. Section IV

Received 20 December 1999; revision received 30 June 2000; accepted for publication 5 October 2000.

*Ph.D. Student, School of Electrical and Electronic Engineering.
†Professor, School of Electrical and Electronic Engineering, Block S2; ensundara@ntu.edu.sg.
‡Associate Professor, School of Electrical and Electronic Engineering.

presents a comparison study of using a fully tuned RBFN and the RBFN with only tuning of the weights based on an F8 longitudinal aircraft model. Section V discusses the dynamic structured RBFN and presents simulation results. Section VI summarizes the conclusions from this study.

II. Problem Formulation

The aircraft dynamics is represented by a bounded-input/bounded-output continuous system,

$$\Sigma: \dot{\boldsymbol{x}} = \boldsymbol{f}(\boldsymbol{x}, \boldsymbol{u}) \tag{1}$$

In Eq. (1), smoothness of $\boldsymbol{f}(\,)$ is assumed, $\boldsymbol{x}$ is an $n \times 1$ state vector, and $\boldsymbol{u}$ is a $p \times 1$ control vector. It is assumed that $\boldsymbol{f}(0, 0) = 0$, so that the origin is an equilibrium state. In general, the number of control inputs is less than that of the states ($p < n$). Under this condition, only the p states can be tracked perfectly. When $\boldsymbol{x}$ is partitioned into $\boldsymbol{x}_t$ and $\boldsymbol{x}_r$, the aircraft dynamics can be written as

$$\Sigma: \begin{bmatrix} \dot{\boldsymbol{x}}_t \\ \dot{\boldsymbol{x}}_r \end{bmatrix} = \begin{bmatrix} \boldsymbol{f}_t(\boldsymbol{x}, \boldsymbol{u}) \\ \boldsymbol{f}_r(\boldsymbol{x}, \boldsymbol{u}) \end{bmatrix} \tag{2}$$

The problem studied in this paper is to design a neurocontroller such that the given p states $\boldsymbol{x}_t \in \boldsymbol{x}$ of the aircraft can track the desired commands $\boldsymbol{x}_{\mathrm{dt}}$ accurately, while the other states $\boldsymbol{x}_r \in \boldsymbol{x}$ asymptotically approach the equilibrium point $\boldsymbol{x}_{\mathrm{dr}}$. Determination of the conditions under which a solution $\boldsymbol{u}_{\mathrm{d}}(t)$ exists forms part of the theoretical control problem. Based on the earlier studies in Ref. 19, to achieve the preceding control target, the system should satisfy Assumption 1.

Assumption 1: Under Assumption 1, $\boldsymbol{f}_t(\boldsymbol{x}, \boldsymbol{u})$ has a continuous bounded partial derivative in a certain neighborhood of all of the points along the desired trajectory, and the matrix

$$\frac{\partial \boldsymbol{f}_t(\boldsymbol{x}, \boldsymbol{u})}{\partial \boldsymbol{u}^T}$$

is nonsingular.

It follows from the implicit function theorem that the desired $\boldsymbol{u}_{\mathrm{d}}(t)$ can be expressed as

$$\boldsymbol{u}_{\mathrm{d}}(t) = \bar{\boldsymbol{f}}_t(\boldsymbol{x}_{\mathrm{d}}, \dot{\boldsymbol{x}}_{\mathrm{dt}}) \tag{3}$$

where $\bar{\boldsymbol{f}}_t$, a $p \times 1$ smooth function, is the inverse function of $\boldsymbol{f}_t$, and $\boldsymbol{x}_{\mathrm{d}} = [\boldsymbol{x}_{\mathrm{dt}}^T, \boldsymbol{x}_{\mathrm{dr}}^T]^T$.

In the next section, an RBFN will be chosen to approximate Eq. (3). The main contribution of this paper is using Lyapunov stability theory to derive the tuning rule for updating all of the parameters of the RBFN.

III. Fully Tuned RBFN Controller

The tuning law for adapting the weights of the RBFN has been derived in earlier papers,[7,10] whereas in this paper, the updating rule for a fully tuned RBFN controller is derived based on a feedback-error-learning strategy. The robustness of the proposed algorithm is also analyzed.

A. Control Strategy

A simple control strategy, which is similar to the well-known feedback-error-learning scheme, is proposed in Fig. 1. This scheme is adopted in the study because it has the advantage of generating the desired input signal without requiring that the network be trained initially offline. Moreover, the outputs of the neurocontroller $\boldsymbol{u}_{\mathrm{nn}}$, which has fault tolerant ability through online learning, only depend on the desired input profile.

If all of the states of the system are accessible, the tracking error $\boldsymbol{e}$ is defined as $\boldsymbol{e} = \boldsymbol{x} - \boldsymbol{x}_{\mathrm{d}}$. The error dynamics for the overall system is

$$\dot{\boldsymbol{e}} = \dot{\boldsymbol{x}} - \dot{\boldsymbol{x}}_{\mathrm{d}} = \boldsymbol{f}(\boldsymbol{x}, \boldsymbol{u}) - \boldsymbol{f}(\boldsymbol{x}_{\mathrm{d}}, \boldsymbol{u}_{\mathrm{d}}) \tag{4}$$

Because this paper is only concerned with the local stability of the system, Eq. (4) can be expanded along the desired trajectory using a Taylor series. When all of the higher-order terms are neglected,

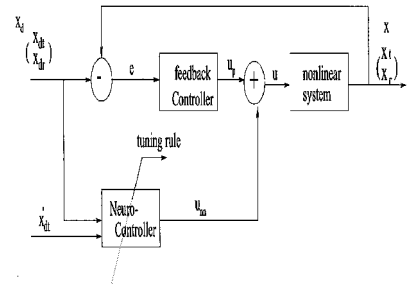


Fig. 1 Control structure of NN–proportional-derivative controller.

$$\dot{\boldsymbol{e}} = \left.\frac{\partial \boldsymbol{f}(\boldsymbol{x}, \boldsymbol{u})}{\partial \boldsymbol{x}^T}\right|_{x = x_{\mathrm{d}}, u = u_{\mathrm{d}}} (\boldsymbol{x} - \boldsymbol{x}_{\mathrm{d}}) + \mathcal{O}(\boldsymbol{x} - \boldsymbol{x}_{\mathrm{d}}) + \left.\frac{\partial \boldsymbol{f}(\boldsymbol{x}, \boldsymbol{u})}{\partial \boldsymbol{u}^T}\right|_{x = x_{\mathrm{d}}, u = u_{\mathrm{d}}} (\boldsymbol{u} - \boldsymbol{u}_{\mathrm{d}}) + \mathcal{O}(\boldsymbol{u} - \boldsymbol{u}_{\mathrm{d}}) \tag{5}$$

where $\mathcal{O}(\,)$ represents higher-order terms. By the substitution of $\partial \boldsymbol{f}(\boldsymbol{x}, \boldsymbol{u})/\partial \boldsymbol{x}^T|_{x = x_{\mathrm{d}}, u = u_{\mathrm{d}}}$ by $A(t)$ and $\partial \boldsymbol{f}(\boldsymbol{x}, \boldsymbol{u})/\partial \boldsymbol{u}^T|_{x = x_{\mathrm{d}}, u = u_{\mathrm{d}}}$ by $B(t)$, and by neglecting all of the higher-order terms,

$$\dot{\boldsymbol{e}} \simeq A(t)(\boldsymbol{x} - \boldsymbol{x}_{\mathrm{d}}) + B(t)(\boldsymbol{u} - \boldsymbol{u}_{\mathrm{d}}) \tag{6}$$

With the widely used technology of the control configured vehicle, the basic airframe may be designed to have low or even negative static stability in certain flight regimes. The augmented stability is then implemented by considering the controller's dynamics. In this study, a conventional controller is used in the strategy to improve the stability and response of the closed-loop system. This conventional controller can be realized by any approach like pole placement, linear quadratic regulator, H_∞ controller, and so on. To illustrate the concept, a proportional controller is designed that satisfies Assumption 2.

Assumption 2: The closed-loop dynamic system, whose feedback controller is designed based on the nominal aircraft model, is stable along its desired flight trajectory.

With only the linear proportional controller $\boldsymbol{u} = K_p \boldsymbol{e}$, the error dynamics is

$$\dot{\boldsymbol{e}} = [A(t) + B(t)K_p]\boldsymbol{e} - B(t)\boldsymbol{u}_{\mathrm{d}} \tag{7}$$

Although the system is stable according to Assumption 2, the tracking performance deteriorates greatly because of the existence of $B(t)\boldsymbol{u}_{\mathrm{d}}$. Therefore, the RBFN controller is used for online learning of $\boldsymbol{u}_{\mathrm{d}}$. Thus, the total control signal to the aircraft is the sum of the proportional controller and the RBFN controller,

$$\boldsymbol{u} = K_p \times \boldsymbol{e} + \boldsymbol{u}_{\mathrm{nn}} \tag{8}$$

B. RBFN Parameterization

When the RBFN's inputs ζ are set to $\zeta = [\boldsymbol{x}_{\mathrm{d}}^T, \dot{\boldsymbol{x}}_{\mathrm{dt}}^T]^T$, $\boldsymbol{u}_{\mathrm{d}}$ can be approximated by an RBFN through online learning,

$$\boldsymbol{u}_{\mathrm{d}} = \boldsymbol{u}_{\mathrm{nn}}^* + \epsilon_h = \sum_{k=1}^{h} \boldsymbol{w}_k^* \exp\left(-\frac{1}{\sigma_k^{*2}} \left\| \zeta - \mu_k^* \right\|^2\right) + \epsilon_h = W^{*T} \phi(\mu^*, \sigma^*, \zeta) + \epsilon_h \tag{9}$$

where W^* is an $h \times p$ optimal weight matrix (h indicates the number of hidden neurons), and ϕ is an $h \times 1$ Gaussian function, which is determined by the optimal centers μ^* and widths σ^*. For simplicity, ϕ^* is used instead of $\phi(\mu^*, \sigma^*, \zeta)$ hereafter. With use of approximation theory, the inherent approximation error ϵ_h can be reduced arbitrarily by increasing the number of h (Ref. 20). It is then reasonable to assume that ϵ_h is bounded by a constant ϵ_H, and

$$\epsilon_H = \sup_{t \in R^+} |\epsilon_h(t)| \tag{10}$$

If the variables ($\boldsymbol{x}_{\mathrm{d}}$ and $\dot{\boldsymbol{x}}_{\mathrm{dt}}$) are in the compact sets, ϵ_H is finite. If they are not in compact sets, a scaling mechanism can be utilized to convert them into a compact set.[21]

With the RBFN controller, according to Eq. (8), the control input vector $\boldsymbol{u}$ is

$$\boldsymbol{u} = \sum_{k=1}^{h} \hat{\boldsymbol{w}}_k \exp\left(-\frac{1}{\hat{\sigma}_k^2}\|\zeta - \hat{\mu}_k\|^2\right) + K_p \boldsymbol{e} = \hat{W}^T \hat{\phi} + K_p \boldsymbol{e} \quad (11)$$

where $\hat{\boldsymbol{w}}_k$ is the estimated weights, $\hat{\sigma}_k$ and $\hat{\mu}_k$ are the estimated center and width for the kth hidden neuron, and $\hat{W}^T$ and $\hat{\phi}$ are the corresponding matrix expressions. By the substitution of Eqs. (9) and (11) into Eq. (6), the error dynamics become,

$$\dot{\boldsymbol{e}} = [A(t) + B(t)K_p]\boldsymbol{e} + B(t)\left(\hat{W}^T\hat{\phi} - W^{*T}\phi^* - \epsilon_h\right) \quad (12)$$

When $J(t) = A(t) + B(t)K_p$, is used, the definition of $\tilde{W}$ is the difference between W^* and $\hat{W}$; $\tilde{\phi}$ is the difference between ϕ^* and $\hat{\phi}$; and, neglecting higher-order item $\tilde{W}\tilde{\phi}$, the error dynamics may be written as

$$\dot{\boldsymbol{e}} \doteq J\boldsymbol{e} - B(t)(\hat{W}^T\tilde{\phi} + \tilde{W}^T\hat{\phi}) - B(t)\epsilon_h \quad (13)$$

where $B(t)(\hat{W}^T\tilde{\phi} + \tilde{W}^T\hat{\phi})$ represents the learning error E_l.

C. Stable Adaptive Law for a Fully Tuned RBFN

To derive the stable tuning law, choose the following Lyapunov function:

$$V = \tfrac{1}{2}\boldsymbol{e}^T P\boldsymbol{e} + \tfrac{1}{2}\operatorname{tr}(\tilde{W}^T\Theta\tilde{W}) + \tfrac{1}{2}\tilde{\phi}^T\Lambda\tilde{\phi} \quad (14)$$

where P is an $n \times n$ symmetric positive definite matrix, and Θ and Λ are $h \times h$ nonnegative definite matrices. The derivative of the Lyapunov function is given by

$$\dot{V} = \tfrac{1}{2}[\dot{\boldsymbol{e}}^T P\boldsymbol{e} + \boldsymbol{e}^T P\dot{\boldsymbol{e}}] + \operatorname{tr}(\tilde{W}^T\Theta\dot{\tilde{W}}) + \tilde{\phi}^T\Lambda\dot{\tilde{\phi}} \quad (15)$$

By the substitution of Eq. (13) into Eq. (15),

$$\dot{V} = -\boldsymbol{e}^T Q\boldsymbol{e} - \epsilon_h^T B(t)^T P\boldsymbol{e} - \tilde{\phi}^T\hat{W}B(t)^T P\boldsymbol{e} - \hat{\phi}^T\tilde{W}B(t)^T P\boldsymbol{e} + \operatorname{tr}(\tilde{W}^T\Theta\dot{\tilde{W}}) + \tilde{\phi}^T\Lambda\dot{\tilde{\phi}} \quad (16)$$

where $Q = -\frac{1}{2}(J^T P + PJ)$. Because J is Hurwitz, the Lyapunov function can always be found and has a unique solution. Noting in Eq. (16) that

$$\operatorname{tr}(\tilde{W}^T\Theta\dot{\tilde{W}}) = \sum_{i=1}^{p} \tilde{\boldsymbol{w}}_i^T\Theta\dot{\tilde{\boldsymbol{w}}}_i \quad (17)$$

$$\hat{\phi}^T\tilde{W}B(t)^T P\boldsymbol{e} = \sum_{i=1}^{p} \hat{\phi}^T\tilde{\boldsymbol{w}}_i\boldsymbol{B(t)}_i^T P\boldsymbol{e} \quad (18)$$

Eq. (16) becomes

$$\dot{V} = -\boldsymbol{e}^T Q\boldsymbol{e} - \epsilon_h^T B(t)^T P\boldsymbol{e} + \tilde{\phi}^T[-\hat{W}B(t)^T P\boldsymbol{e} + \Lambda\dot{\tilde{\phi}}] + \sum_{i=1}^{p}\left[-\tilde{\boldsymbol{w}}_i^T\hat{\phi}B(t)_i^T P\boldsymbol{e} + \tilde{\boldsymbol{w}}_i^T\Theta\dot{\tilde{\boldsymbol{w}}}_i\right] \quad (19)$$

where $\dot{\tilde{\boldsymbol{w}}}_i$ is the ith column of matrix $\dot{\tilde{W}}$ and $B(t)_i$ is the ith column of matrix $B(t)$.

If $\dot{\tilde{\boldsymbol{w}}}_i$ and $\dot{\tilde{\phi}}$ are selected as

$$\dot{\tilde{\boldsymbol{w}}}_i = \Theta^{-1}\hat{\phi}B(t)_i^T P\boldsymbol{e}, \qquad i = 1, \ldots, p \quad (20)$$

$$\dot{\tilde{\phi}} = \Lambda^{-1}\hat{W}B(t)^T P\boldsymbol{e} \quad (21)$$

then Eq. (19) becomes

$$\dot{V} = -\boldsymbol{e}^T Q\boldsymbol{e} - \epsilon_h^T B(t)^T P\boldsymbol{e} \quad (22)$$

$\dot{V}$ can be demonstrated nonpositive according to Eq. (10),

$$\dot{V} \le -\|\boldsymbol{e}\|\lambda_{\min}(Q)\|\boldsymbol{e}\| + \|\boldsymbol{e}\|\|P\|\|B(t)\|\epsilon_H \quad (23)$$

Let $\|B(t)\|\epsilon_H = \delta_{\epsilon_h}$. It can then be derived directly that $\dot{V}$ is nonpositive when

$$\|\boldsymbol{e}\| \ge \frac{\|P\|}{\lambda_{\min}(Q)}\delta_{\epsilon_h} = E_a \quad (24)$$

With the use of the universal approximation proposition,[20] by increasing the number h, ϵ_H can be reduced arbitrarily small, which means that E_a tends to zero when $h \to \infty$, and the negativeness of the Lyapunov function can be guaranteed, resulting in the stability of the overall system.

Because $\dot{\tilde{\boldsymbol{w}}}_i = \dot{\boldsymbol{w}}_i^* - \dot{\hat{\boldsymbol{w}}}_i$, $\dot{\tilde{\phi}} = \dot{\phi}^* - \dot{\hat{\phi}}$, and $\dot{\boldsymbol{w}}_i^* = \boldsymbol{0}$ and $\dot{\phi}^* = \boldsymbol{0}$, the tuning rules are

$$\dot{\hat{\boldsymbol{w}}}_i = -\Theta^{-1}\hat{\phi}B(t)_i^T P\boldsymbol{e}, \qquad i = 1, \ldots, p \quad (25)$$

$$\dot{\hat{\phi}} = -\Lambda^{-1}\hat{W}B(t)^T P\boldsymbol{e} \quad (26)$$

D. Robustness Analysis

From the preceding derivation, it can be seen that using the proposed method, provided Assumption 2 stands, the nonlinear matrix $A(t)$ and $B(t)$ need not be known exactly. However, because the tuning rules derived involve the prior knowledge of matrix $B(t)$, the robustness of the tuning laws analyzed in case $B(t)$ is partially known or unknown (actuator fail or model error).

In case 1, matrix $B(t)$ varies, but satisfies $B(t) = k(t)B_0$, where B_0 is a known nominal value. In case 1, $k(t)$ is a positive scalar varying with time.

Provided the weight tuning rule and the tuning rule of ϕ are given separately as

$$\dot{\hat{\boldsymbol{w}}}_i = -k\Theta^{-1}\hat{\phi}B_{0i}^T P\boldsymbol{e}, \qquad i = 1, \ldots, p \quad (27)$$

$$\dot{\hat{\phi}} = -k\Lambda^{-1}\hat{W}B_0^T P\boldsymbol{e} \quad (28)$$

$\dot{V}$ can be demonstrated negative using the same condition as in Eq. (24). In implementation, k can be replaced by a positive scalar η. Because η and k have the same sign, the system's convergence characteristic will not change.

In case 2, $B(t)$ is unknown and is represented by a nominal value plus its variation, $B(t) = B_0[I + \Delta B(t)]$.

Assume that $\|\Delta B(t)\| \le M$. M is the upper bound for the uncertainty. The derivative of Lyapunov function of Eq. (15) is reanalyzed and is given by

$$\dot{V} = -\boldsymbol{e}^T Q\boldsymbol{e} - \boldsymbol{e}^T P[B_0 + B_0\Delta B(t)]\epsilon - \left[\tilde{\phi}^T\hat{\boldsymbol{w}}B_0\Delta B(t)P\boldsymbol{e} + \hat{\phi}^T\tilde{W}B_0\Delta B(t)P\boldsymbol{e}\right] + \tilde{\phi}^T\left\{-\hat{W}[B_0 + B_0\Delta B(t)]^T P\boldsymbol{e} + \Lambda\dot{\tilde{\phi}}\right\} + \sum_{i=1}^{p}\left[-\tilde{\boldsymbol{w}}_i^T\hat{\phi}(B_{0i} + B_{0i}\Delta B_i)^T P\boldsymbol{e} + \tilde{\boldsymbol{w}}_i{}^T\Theta\dot{\tilde{\boldsymbol{w}}}_i\right] \quad (29)$$

When the weight tuning rule and the tuning rule of ϕ using B_0 are rewritten,

$$\dot{\hat{\boldsymbol{w}}}_i = -\Theta^{-1}\hat{\phi}B_{0i}^T P\boldsymbol{e}, \qquad i = 1, \ldots, p \quad (30)$$

$$\dot{\hat{\phi}} = -\Lambda^{-1}\hat{W}B_0^T P\boldsymbol{e} \quad (31)$$

then

$$\dot{V} = -\boldsymbol{e}^T Q\boldsymbol{e} - \boldsymbol{e}^T P[B_0 + B_0\Delta B(t)]\epsilon_h - (\tilde{\phi}^T\hat{\boldsymbol{w}} + \hat{\phi}^T\tilde{W})B_0\Delta B(t)P\boldsymbol{e} \quad (32)$$

Reanalyzing $\dot{V}$, we have

$$\dot{V} \le -\|\boldsymbol{e}\|\lambda_{\min}(Q)\|\boldsymbol{e}\| + \|\boldsymbol{e}\|\|P\|\|B_0\|\|I + \Delta B(t)\|\epsilon_H + \|\tilde{\Phi}^T\hat{W} + \hat{\Phi}^T\tilde{W}\|\|B_0\|\|\Delta B(t)\|\|P\|\|e\| \quad (33)$$

Because $\|I + \Delta B(t)\| \epsilon_H \le \delta'_{\epsilon_h}$, and defining $\|(\tilde{\phi}^T \hat{W} + \hat{\phi}^T \tilde{W})\| \|P\| = \zeta(\tilde{\phi}, \tilde{W})$ to be brief, it can be derived directly that $\dot{V}$ is nonpositive when

$$\|\boldsymbol{e}\| \ge \frac{\|P\|\|B_0\|}{\lambda_{\min}(Q)} \delta'_{\epsilon_h} + \frac{\|P\|\|B_0\|}{\lambda_{\min}(Q)} \zeta(\tilde{\phi}, \tilde{W}) \|\Delta B(t)\| = E_a + E_{\text{lm}} \tag{34}$$

In the preceding equation, E_a is caused by the approximation inaccuracy ϵ_h, and E_{lm} is caused by the product of the learning error $E_l(\|(\tilde{\phi}^T \hat{W} + \hat{\phi}^T \tilde{W})\|)$ and modeling error $E_m(\|\Delta B(t)\|)$.

The following remarks can be made:

1) If there is neither approximation error, that is, $\epsilon_h = 0$, nor model error, that is, $\Delta B(t) = 0$, from Eq. (33), $\dot{V}$ is negative semidefinite, and, hence, the stability of the overall scheme is guaranteed.

2) The approximation error E_a is related to the number of hidden neurons used in the RBFN. Given enough hidden units, this number converges to a very small number.

3) The second error item E_{lm} is a combination of the learning error and modeling error. The product of ζ and $\Delta B(t)$ indicates that, even under large model error $\Delta B(t)$, if the RBFN learns the desired value W^* and ϕ^* correctly, E_{lm} is still very small.

4) Matrix $B(t)$ plays a very important role in the strategy. It is better to use some strategies to identify this matrix or incorporate a robust control strategy to obtain good performance.

E. Implementation of the Tuning Rule

In Eq. (26), the Gaussian function $\hat{\phi}$ is embedded with the centers' locations $\hat{\mu}$ and widths $\hat{\sigma}$, that is, $\hat{\phi} = \phi(\hat{\mu}, \hat{\sigma}, \zeta)$. When all of the adaptable parameters are combined into a big vector, $\chi = [\hat{\boldsymbol{w}}_1^T, \hat{\mu}_1^T, \hat{\sigma}_1, \ldots, \hat{\boldsymbol{w}}_h^T, \hat{\mu}_h^T, \hat{\sigma}_h]^T$, a simple updating rule for χ can be derived. Because the real implementation is carried out in a discrete-time framework, the updating laws are also derived under discrete form.

First, Eq. (25) can be converted into

$$\dot{\hat{\boldsymbol{w}}}_i = -\Theta^{-1} \hat{\phi} B_i^T P \boldsymbol{e}, \qquad i = 1, \ldots, p$$

$$\Rightarrow \dot{\hat{W}}^T = -B(t)^T P \boldsymbol{e} \hat{\phi}^T \Rightarrow \dot{\hat{\boldsymbol{w}}}_j = -B(t)^T P \boldsymbol{e} \hat{\phi}_j \qquad j = 1, \ldots, h, \qquad \Theta = I \tag{35}$$

where $\hat{\boldsymbol{w}}_j$, a column vector, is the jth row of matrix $\hat{W}$. Define $\hat{g} = \hat{W}^T \hat{\phi}$ as the output of the RBFN. Then, $\hat{\phi}$ is the derivative of $\hat{\boldsymbol{g}}(\,)$ to the weights $\hat{\boldsymbol{w}}$. This equation can be converted into a discrete form,

$$\hat{\boldsymbol{w}}_i(n+1) = \hat{\boldsymbol{w}}_i(n) - \eta_1 \frac{\partial \hat{\boldsymbol{g}}}{\partial \hat{\boldsymbol{w}}_i^T} B(t)^T P \boldsymbol{e}(n), \qquad i = 1, \ldots, h \tag{36}$$

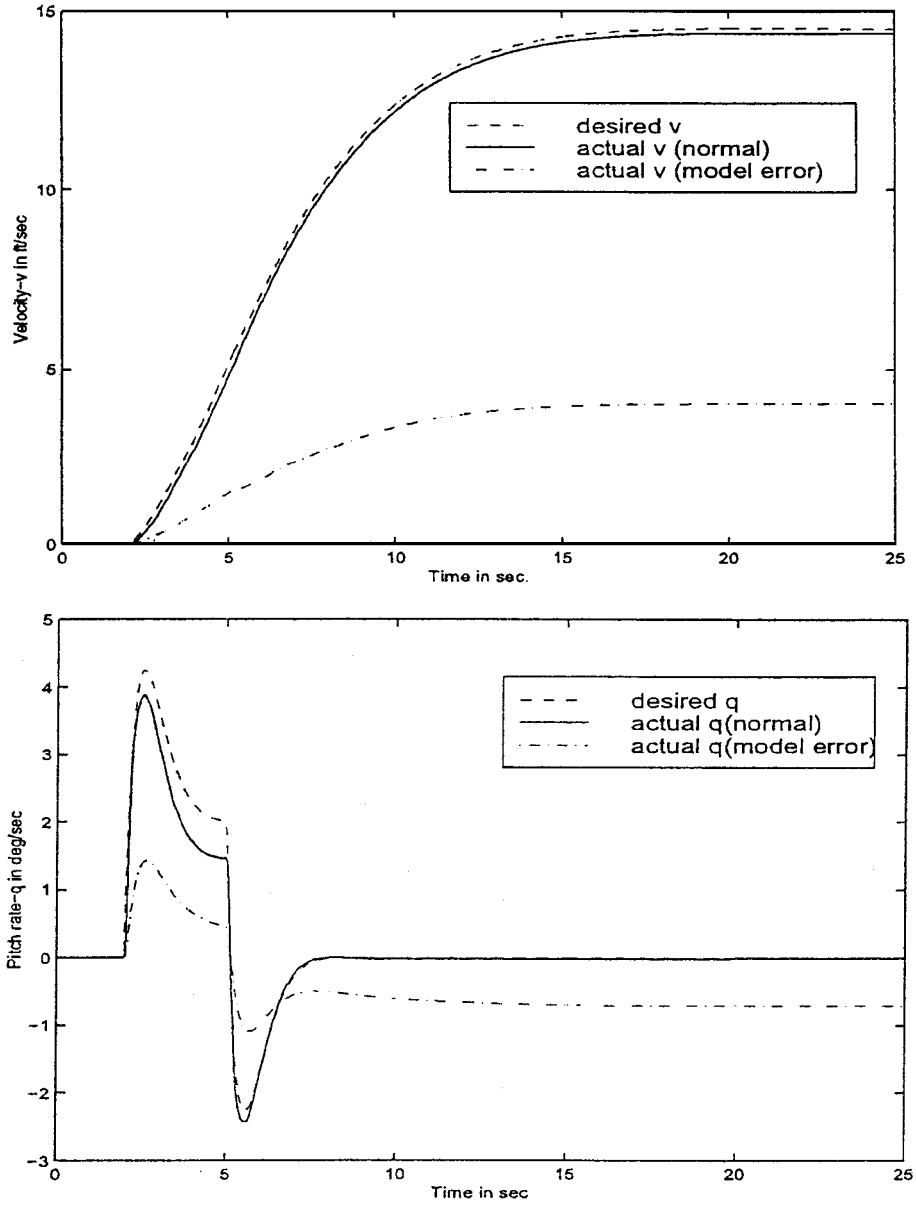


Fig. 2 Desired trajectory and traditional controller response.

Next, from Eq. (26),

$$\Delta\hat{\phi}(n) = \hat{\phi}(n+1) - \hat{\phi}(n) = -\eta_2 \Lambda^{-1} \hat{W} B(t)^T P \boldsymbol{e}(n) \tag{37}$$

As mentioned, the Gaussian function $\hat{\phi} = \phi(\hat{\mu}, \hat{\theta}, \zeta)$ and the RBFN's output $\hat{\boldsymbol{g}} = \hat{W}^T \hat{\phi}$; therefore, the updating laws for centers and widths of the basis functions become

$$\hat{\mu}_i(n+1) = \hat{\mu}_i(n) + \eta_3 \frac{\partial \hat{\phi}(n)}{\partial \hat{\mu}_i^T} \Delta\hat{\phi}(n) = \hat{\mu}_i(n) - \eta_2 \eta_3 \frac{\partial \hat{\phi}(n)}{\partial \hat{\mu}_i^T} \hat{W} B(t)^T P \boldsymbol{e}(n) = \hat{\mu}_i(n) - \eta_2 \eta_3 \frac{\partial \hat{\boldsymbol{g}}}{\partial \hat{\mu}_i^T} B(t)^T P \boldsymbol{e}(n) \qquad i = 1, \ldots, h \tag{38}$$

$$\hat{\sigma}_i(n+1) = \hat{\sigma}_i(n) + \eta_4 \frac{\partial \hat{\phi}(n)}{\partial \hat{\sigma}_i^T} \Delta\hat{\phi}(n) = \hat{\sigma}_i(n) - \eta_2 \eta_4 \frac{\partial \hat{\phi}(n)}{\partial \hat{\sigma}_i^T} \hat{W} B(t)^T P \boldsymbol{e}(n) = \hat{\sigma}_i(n) - \eta_2 \eta_4 \frac{\partial \hat{\boldsymbol{g}}}{\partial \hat{\sigma}_i^T} B(t)^T P \boldsymbol{e}(n) \qquad i = 1, \ldots, h \tag{39}$$

In the preceding equations, η_1, η_2, η_3, and η_4 are positive scalars to be selected properly. Integrating Eqs. (36), (38), and (39) by using the vector $\boldsymbol{\chi}$, the updating rule is

$$\boldsymbol{\chi}(n+1) = \boldsymbol{\chi}(n) - \eta \alpha(n) B(t)^T P \boldsymbol{e}(n) \tag{40}$$

where η is learning rate. Here, $\alpha(n) = \nabla_{\chi} \hat{\boldsymbol{g}}(\zeta_n)$ is the gradient of the function $\hat{\boldsymbol{g}}(\,)$ with respect to the parameter vector $\boldsymbol{\chi}$ evaluated at $\boldsymbol{\chi}(n-1)$, and

$$\alpha(n) = \left[\hat{\phi}_1, \hat{\phi}_1\left(2\hat{\boldsymbol{w}}_1 / \hat{\sigma}_1^2\right)(\zeta_n - \hat{\mu}_1)^T, \hat{\phi}_1\left(2\hat{\boldsymbol{w}}_1 / \hat{\sigma}_1^3\right) \|\zeta_n - \hat{\mu}_1\|^2, \ldots, \hat{\phi}_h, \hat{\phi}_h\left(2\hat{\boldsymbol{w}}_h / \hat{\sigma}_h^2\right)(\zeta_n - \hat{\mu}_h)^T, \hat{\phi}_h\left(2\hat{\boldsymbol{w}}_h / \hat{\sigma}_h^3\right) \|\zeta_n - \hat{\mu}_h\|^2\right]^T \tag{41}$$

Because the number of hidden neurons of the RBFN is fixed in this scheme, the approximation error E_a is quite large at the initial learning period. If $\|\boldsymbol{e}\| < E_a$, then it is possible that $\dot{V} > 0$, which implies that the parameters of the network may drift to infinity. A common way to avoid this problem and to ensure the convergence

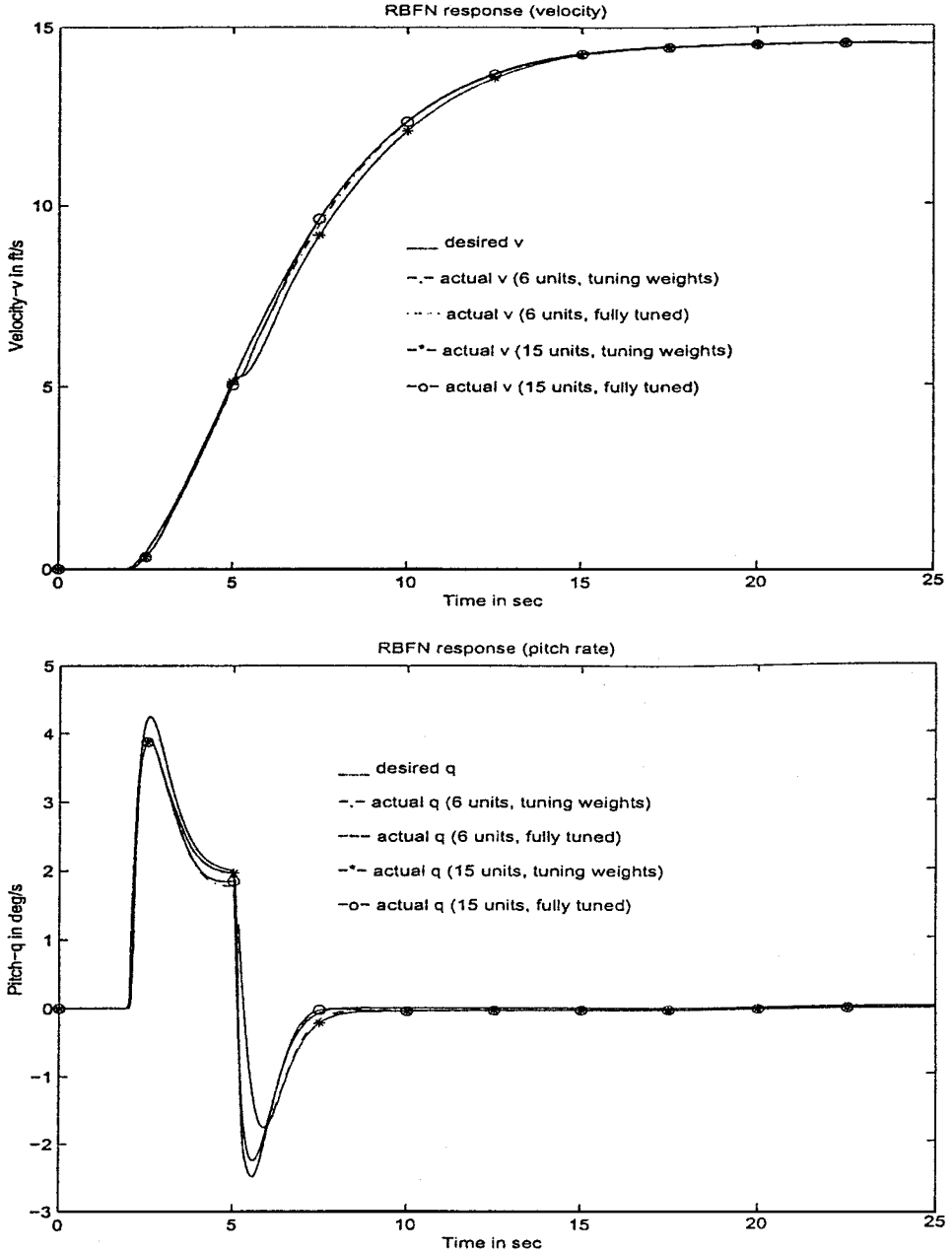


Fig. 3 RBFN response under normal condition.

of approximation error is to incorporate a dead zone in the tuning rules.[7] The tuning rule is given now as

$$\chi(n+1)=\begin{cases}\chi(n)-\eta\alpha(n)\boldsymbol{e}(n) & \text{if } \|\boldsymbol{e}\|\geq e_0, \quad \|\boldsymbol{w}\|\leq h^{\frac{1}{2}}M\\ \chi(n) & \text{otherwise}\end{cases} \tag{42}$$

In Eq. (42), M is a positive scalar, $h^{1/2}M$ is an upper bound on $\|W\|$ (Euclidean norm of weight vector), and e_0 is selected as the required accuracy on the state error $\boldsymbol{e}$. According to Eq. (24), the size of the dead zone should be set to E_a, so that if the error $\|\boldsymbol{e}\| < E_a$, the tuning is stopped and the parameters will not drift away. Unfortunately, because ϵ_H is time varying and unknown, the exact value for E_a can not be estimated; therefore, e_0 is used in the place of $E_a(t)$. If $e_0 > E_a$, then $\dot{V}$ is always negative. If $e_0 < E_a$, when the error $\|\boldsymbol{e}\|$ converges to $e_0 < \|\boldsymbol{e}\| < E_a$, then $\dot{V}$ may be positive, and in this case the upper bound $h^{1/2}M$ is used to prevent the parameter from drifting away.

IV. Simulation Results

In Ref. 4, Sadhukhan and Fetieh presented an exact inverse neurocontroller for the linearized longitudinal dynamics of an F8 aircraft model, $\dot{x} = A\boldsymbol{x}+B\boldsymbol{u}$. The linearized longitudinal model of the F8 aircraft is obtained under the following trim condition: $v(0) = 650$ ft/s (Mach number 0.6), $\alpha(0) = 0.078$ rad, and altitude 20,000 ft. The state vector $\boldsymbol{x}$ comprises the velocity v, angle of attack α, pitch rate q, and pitch angle θ. The control inputs $\boldsymbol{u}$ include the elevator's output δ_{ea} and the throttle's output δ_{ta}. Physical constraints on the elevator are defined as $-26.5 \leq \delta_{\text{ea}} \leq 6.75$ deg. Simulation studies show that the controller provided stable and near desired response in the presence of modeling uncertainty up to 30% (all of the elements of matrices A and B are changed by 30%). However, under 40% model error, the velocity oscillates in the first few seconds, though eventully settling.

With the traditional feedback controller and the RBFN controller's online learning ability, it is shown that the proposed scheme can tolerate more serious fault conditions. To assess the performance of the neuro-flight-controller, the model errors are simulated by changing all of the elements of the state matrix A and control matrix B by 70%. Although simulation studies show that the neurocontroller can also be applied to control the system even when model error is greater than 70%, the results indicate that elevator actuator saturation will be the limiting factor, resulting in deterioration of the

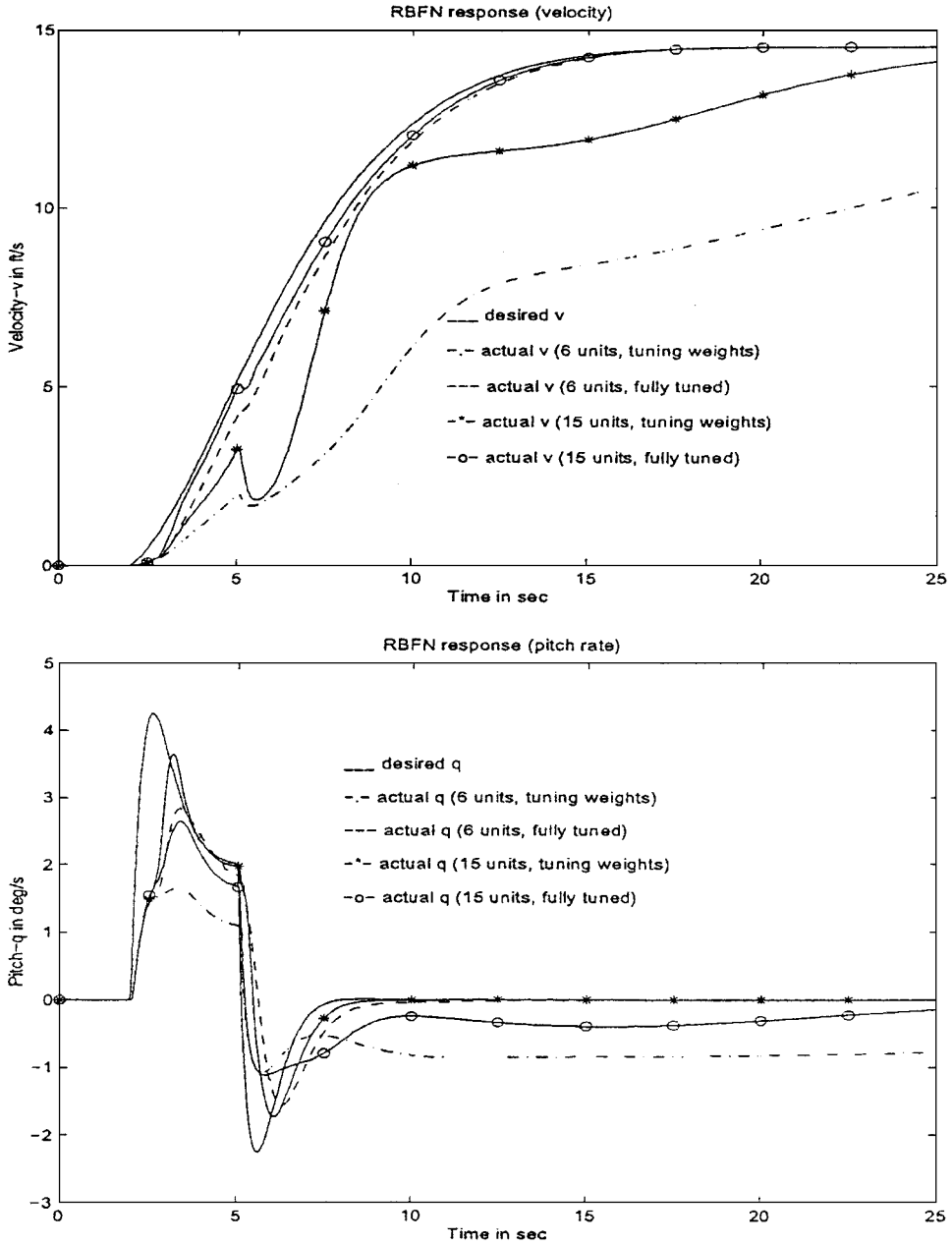


Fig. 4 RBFN response under 70% model error.

tracking performance. The proportional controller gain was selected as

$$K_p = \begin{bmatrix} -0.01 & 0.002 & 2.92 & 0.02 \\ -2.02 & 0 & -0.12 & -0.30 \end{bmatrix}$$

to satisfy Assumption 2, and good performance is observed in nominal condition from Fig. 2. However, from Fig. 2, it can be seen that by using the traditional controller alone, in the case of the 70% model error, the tracking results deteriorate greatly. Hence, the RBFN controller is added in cascade with the aircraft, to improve the tracking accuracy by learning the inverse dynamics of the system. The aim of the study is to compare the tracking accuracy of velocity and pitch rate using a fully tuned RBFN and a traditional RBFN with fixed centers and widths.

At the start of the online learning, according to the a priori knowledge, the number of hidden units are frozen to 6 and 15 separately. The centers of the hidden units are fixed with a grid method described in Ref. 21, and their widths are set to 0.5. Figure 3 plots the tracking performance under the normal case, whereas Fig. 4 is under 70% model error. Figure 3 indicates that under the nominal condition, because of the existence of the predesigned feedback controller, all of the strategies can achieve good performance. However, when there is 70% model error, a better tracking result is observed using fully tuned RBFN. From Fig. 4, it can be easily seen that, using fully tuned RBFN, the tracking performance is quite good with only six hidden units, whereas if only the weights of the RBFN are updated, it is obvious that the controller can not track the desired commands correctly. Although increasing the number of hidden units to 15 improves the performance, it is still not as good as the RBFN with all of the parameters tunned. Figure 5 depicts the evolution of tracking errors of both velocity and pitch rate. A detailed quantitative comparison in terms of the average $E(\text{av})$ and peak tracking errors $E(\max)$ for different sized RBFN is given in Table 1 (all of the results are compared under 70% model error). This phenomenon indicates that in using a traditional RBFN, a priori knowledge should be accurately embedded into the centers and widths of the hidden units. However, with a fully tuned RBFN, because its centers and widths can be modified during the online learning, this requirement is not so stringent.

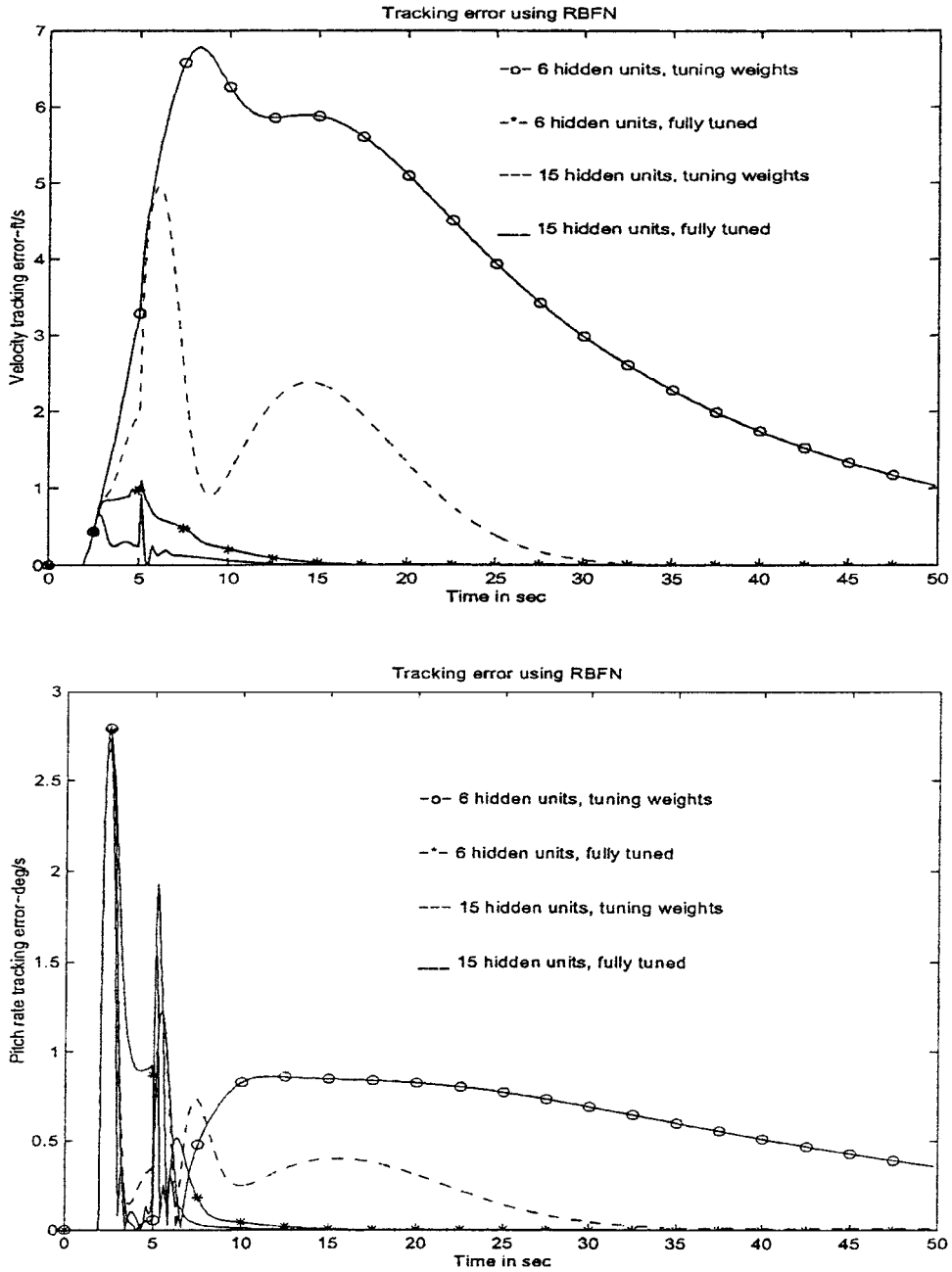


Fig. 5 Evolution of tracking errors under 70% model error.

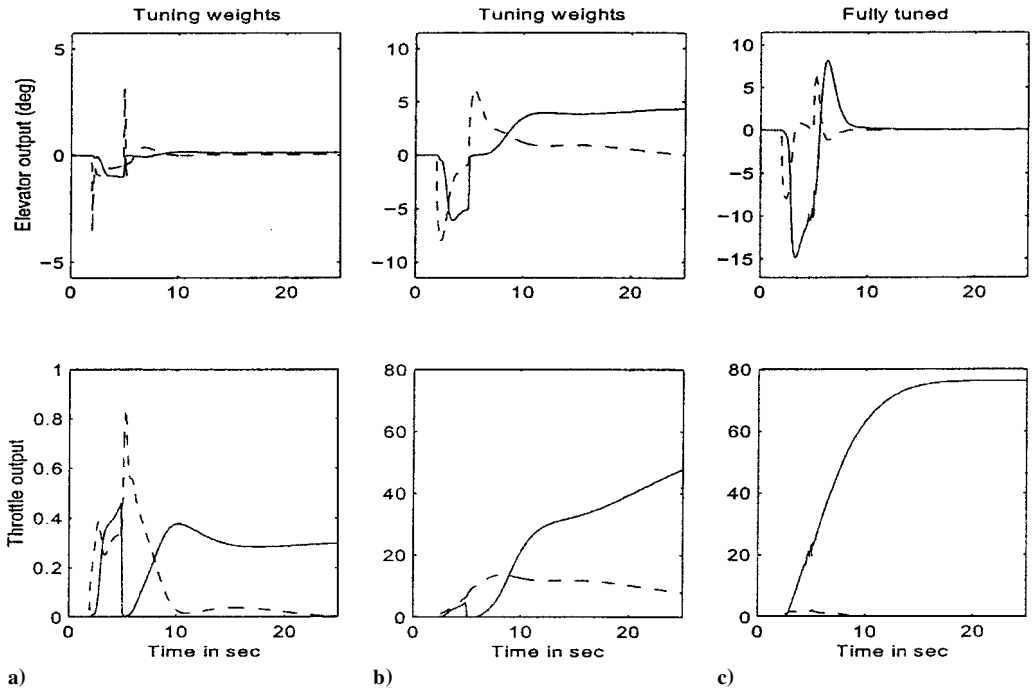


Fig. 6 Control inputs (RBFN with six hidden units): a) normal, b) model error (tuning weights), and c) model error (fully tuned) (the dotted line is the output of the conventional controller, and the solid line refers to the output of the RBFN controller).

Table 1 Tracking results comparison using RBFN

Network structure	Adapted parameters	Hidden units	Tracking error			
			E_v(max)	E_v(av)	E_q(max)	E_q(av)
RBFN	w_i	6	6.78	3.37	2.79	0.68
RBFN	w_i, μ_i, θ_i	6	1.02	0.11	2.78	0.09
RBFN	$\mathbf{w}_i$	15	4.96	0.83	2.67	0.22
RBFN	$\mathbf{w}_i, \mu_i, \theta_i$	15	0.92	0.042	2.63	0.062

It is shown in Fig. 6 how the inverse dynamics can be tolerated online by the RBFN. The continuous line in Figs. 6 represents the outputs of the RBFN controller, whereas the dotted line represents the conventional controller's outputs. For an RBFN controller with only the weights tuned, the outputs of the two controllers under the nominal condition and model error are given in the left column (Fig. 6a) and middle column (Fig. 6b), respectively. It can be seen from the Figs. 6a and 6b that, in the initial period, the conventional controller contributes a lot to maintain the performance whereas the RBFN's outputs are varying, which indicates a process of learning. In the nominal condition, after 10 s, the steady outputs are all due to the RBFN controller, and the outputs of the feedback controller shrinks to zero, which means that the RBFN has learned the system inverse. However, under model error, the traditional RBFN fails to catch up with the inverse dynamics of the system. This is actually caused by the incorrect centers and widths used. Compared to Fig. 6b, it is shown in the right column (Fig. 6c) that, with a fully tuned RBFN, the changes in the model dynamics can be grasped quickly. As a result, the control inputs generated are mainly formed by the RBFN controller. Note that they are well within the limits without saturation.

V. Dynamic Structured RBFN and Comparison Results

To implement a fully tuned RBFN as discussed in the preceding sections, the number of the hidden neurons should be selected a priori by an ad hoc or trial-and-error procedure. However, in the case of approximating the inverse dynamics of a very complicated system, this procedure will waste a lot of time, and superfluous hidden neurons will be added to the network. To avoid this problem, the performance of the dynamic structured RBFN are investigated. The first are RBFN with growing strategy (GRBFN) and the second are RBFN incorporating both growing and pruning strategies (GAP RBFN).

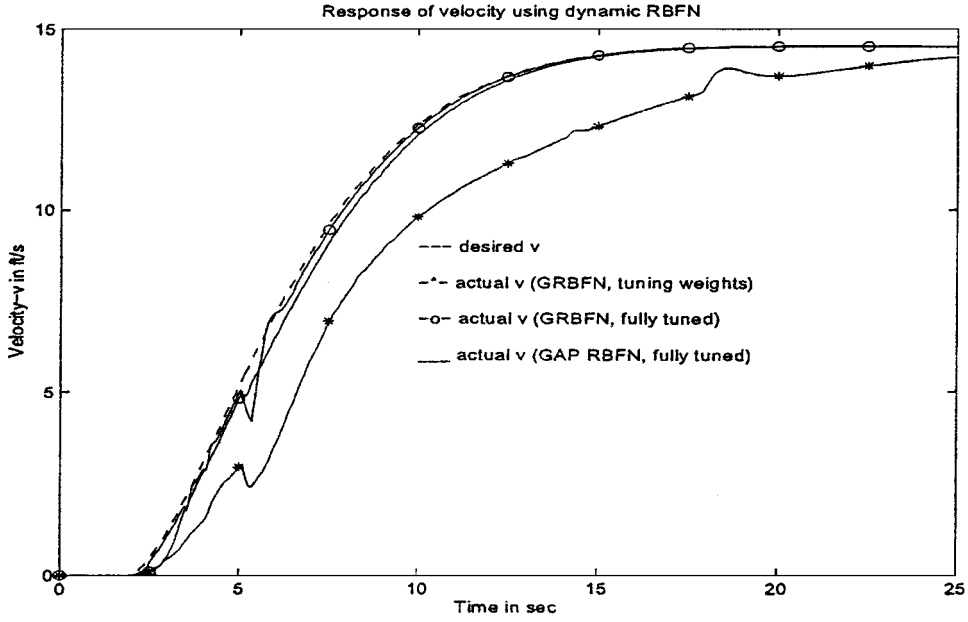
For sequential learning of RBFN, Platt[22] proposed a resource allocating network (RAN), where hidden neurons are added based on the novelty of the new data. Lu et al.[23] extended Platt's algorithm by incorporating a pruning strategy so that those neurons that consistently made little contribution to the network output are removed. The resulting network, known as minimal RAN, was shown to be even more parsimonious for several applications in the areas of function approximation and nonlinear system identification than RAN, with better accuracy. The same GAP strategy as described in Refs. 22 and 23 will be utilized in this section to design the GRBFN controller and GAP RBFN controller. The simulations are intended to compare the tracking accuracy as well as the resulting network complexity using RBFN and dynamic structured RBFN (including GRBFN and GAP RBFN).

Figure 7a shows the tracking result under model error using GRBFN and GAP RBFN separately, and Fig. 7b describes the history for the hidden neurons. From Figs. 7a and 7b, it can be seen that the network gradually learns the system inverse by recruiting new hidden units according to the novelty of the input data. The results also illustrate that, using the fully tuned strategy, the output error is much smaller than the RBFN with only the weights tuned, and the network is more compact (less hidden neurons). The other phenomenon reflected in Figs. 7a and 7b is that, with a GAP RBFN, a more compact network structure can be implemented. Even compared to the fully tuned GRBFN and tracking error, use of the two dynamic structured RBFN gives very close results.

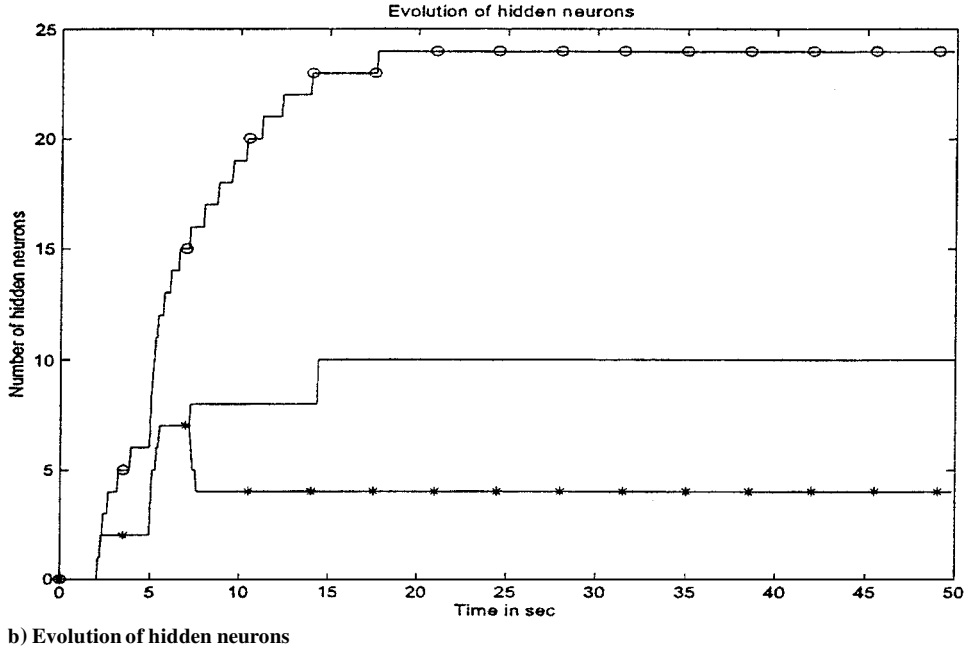
Quantitative comparisons are given in Table 2. It is shown in Table 2 that, using GRBFN, with only the weights tuned, 24 hidden neurons are used and that the average tracking errors for velocity and pitch rate are 0.79 and 0.21, respectively. With fully tuned GRBFN, these numbers decrease to 0.06 and 0.074, and the number of hidden units used is only 10. When compared with fully tuned GRBFN, with a pruning strategy, the GAP RBFN can achieve even better performance, that is, less hidden neurons (only four units are needed) and small tracking error [E_v(av) = 0.026 and E_q(av) = 0.037].

Table 2 Tracking results comparison using different RBFN structures

Network structure	Initialization	Adapted parameters	Hidden units	Tracking error			
				E_v(max)	E_v(av)	E_q(max)	E_q(av)
RBFN	Grid	w_i	15	4.96	0.83	2.67	0.22
RBFN	Grid	w_i, μ_i, θ_i	15	0.92	0.042	2.63	0.062
GRBFN	No need	w_i	24	3.67	0.79	2.75	0.21
GRBFN	No need	w_i, μ_i, θ_i	10	1.69	0.06	2.55	0.074
GAP RBFN	No need	w_i, μ_i, θ_i	4	0.69	0.026	1.75	0.037



a) Dynamic structured RBFN response under model error



b) Evolution of hidden neurons

Fig. 7 System performance using different network structures.

VI. Conclusions

A stable online learning control strategy for a fully tuned RBFN is presented based on the feedback-error-learning scheme, with the aim of improving the tracking accuracy of the control system. The Lyapunov stability theory is utilized to derive a stable tuning rule for updating all of the parameters of the RBFN and to guarantee the local stability of the overall system. Simulation studies based on an F8 aircraft system demonstrate the benefits of using the fully tuned strategy and also confirm the theory. Further studies on the network implementation reveal that dynamic structured RBFN can generate a more compact network structure with a small tracking error, which is especially useful in the practical realization.

References

[1]Calise, A. J., and Rysdyk, R. T., "Nonlinear Adaptive Flight Control Using Neural Networks," *IEEE Control Systems Magazine*, Vol. 18, No. 6, 1998, pp. 14–25.

[2]McFarland, M. B., Rysdyk, R. T., and Calise, A. J., "Robust Adaptive

Control Using Single-Hidden-Layer Feedforward Neural Networks," *Proceedings of the American Control Conference*, IEEE Publications, Piscataway, NJ, 1999, pp. 4178–4182.

[3]Rysdyk, R., Nardi, F., and Calise, A. J., "Robust Adaptive Nonlinear Flight Control Applications Using Neural Networks," *Proceedings of the American Control Conference*, IEEE Publications, Piscataway, NJ, 1999, pp. 2595–2599.

[4]Sadhukhan, D., and Feteih, S., "F8 Neurocontroller Based on Dynamic Inversion," *Journal of Guidance, Control, and Dynamics*, Vol. 19, No. 1, 1996, pp. 150–156.

[5]Napolitano, M. R., Naylor, S., Neppach, C., and Casdorph, V., "Online Learning Nonlinear Direct Neurocontrollers for Restructurable Control Systems," *Journal of Guidance, Control, and Dynamics*, Vol. 18, No. 1, 1995, pp. 170–176.

[6]Kerr, T. H., "Critique of Some Neural Network Architectures and Claims for Control and Estimation," *IEEE Transactions on Aerospace and Electronic Systems*, Vol. 34, No. 2, 1998, pp. 406–419.

[7]Gomi, H., and Kawato, M., "Neural Network Control for a Closed-Loop System Using Feedback-Error Learning," *Neural Networks*, Vol. 6, No. 7, 1993, pp. 933–946.

[8]Gorinevsky, D., Kapitanovsky, A., and Goldenberg, A., "Radial Basis Function Network Architecture for Nonholonomic Motion Planning and Control of Free-Flying Manipulators," *IEEE Transactions on Robotics and Automation*, Vol. 12, No. 3, 1996, pp. 491–496.

[9]Kadirkamanathan, V., and Niranjan, M., "A Function Estimation Approach to Sequential Learning with Neural Networks," *Neural Computation*, Vol. 5, No. 6, 1993, pp. 954–975.

[10]Sanner, R. M., and Slotine, J. J., "Gaussian Networks for Direct Adaptive Control," *IEEE Transactions on Neural Networks*, Vol. 3, No. 6, 1992, pp. 837–863.

[11]Polycarpou, M. M., "Stable Adaptive Neural Control Scheme for Nonlinear Systems," *IEEE Transactions on Automatic Control*, Vol. 41, No. 3, 1996, pp. 447–451.

[12]Polycarpou, M. M., and Mears, M. J., "Stable Adaptive Tracking of Uncertain Systems Using Nonlinearly Parameterized On-Line Approximators, *International Journal of Control*, Vol. 70, No. 3, 1998, pp. 363–384.

[13]Singh, S. N., and Wells, W. R., "Direct Adaptive and Neural Control of Wing-Rock Motion of Slender Delta Wings," *Journal of Guidance, Control, and Dynamics*, Vol. 18, No. 1, 1995, pp. 25–30.

[14]Kim, B. S., and Calise, A. J., "Nonlinear Flight Control Using Neural Networks," *Journal of Guidance, Control, and Dynamics*, Vol. 20, No. 1, 1997, pp. 26–33.

[15]Birgmeier, M., "A Fully Kalman-Trained Radial Basis Function Network for Nonlinear Speech Modeling," *Proceedings of the IEEE International Conference on Neural Networks*, IEEE Publications, Piscataway, NJ, Vol. 1, 1995, pp. 259–264.

[16]Warwick, K., "The Control of Dynamical Systems by Neural Networks," *IEEE/IAS International Conference on Industrial Automation and Control*, IEEE Publications, Piscataway, NJ, 1995, pp. 341–346.

[17]Panchapakesan, C., Ralph, D., and Palaniswami, M., "Effects of Moving the Centers in an RBF Network," *Proceedings of the 1998 IEEE International Joint Conference on Neural Networks*, IEEE Publications, Piscataway, NJ, Vol. 2, 1998, pp. 1256–1260.

[18]Li, Y., Sundararajan, N., and Saratchandran, P., "Neuro-Controller Design for Nonlinear Fighter Aircraft Maneuver Using Fully Tuned RBF Networks," *Automatica* (to be published).

[19]Narendra, K. S., Mukhopadhyay, S., "Adaptive Control Using Neural Networks and Approximate Models," *IEEE Transactions on Neural Networks*, Vol. 8, No. 3, 1997, pp. 475–485.

[20]Park, J., and Sandberg, I. W., "Universal Approximation Using Radial Basis Function Networks," *Neural Computation*, Vol. 3, No. 2, 1991, pp. 246–257.

[21]Fabri, S., Kadirkamanathan, V., "Dynamic Structure Neural Networks for Stable Adaptive Control of Nonlinear Systems," *IEEE Transactions on Neural Networks*, Vol. 7, No. 5, 1996, pp. 1151–1167.

[22]Platt, J. C., "A Resource Allocating Network for Function Interpolation," *Neural Computation*, Vol. 3, No. 2, 1991, pp. 213–225.

[23]Lu, Y. W., Sundararajan, N., and Saratchandran, P., "Performance Evolution of a Sequential Minimal RBF Neural Network Learning Algorithm," *IEEE Transactions on Neural Networks*, Vol. 9, No. 2, 1998, pp. 308–318.